Passage Retrieval for Question Answering using Sliding Windows

Mahboob Alam Khalid
ISLA, University of Amsterdam
mahboob@science.uva.nl

Suzan Verberne
Radboud University Nijmegen
s.verberne@let.ru.nl

Abstract

The information retrieval (IR) community has investigated many different techniques to retrieve passages from large collections of documents for question answering (QA). In this paper, we specifically examine and quantitatively compare the impact of passage retrieval for QA using sliding windows and disjoint windows. We consider two different data sets, the TREC 2002–2003 QA data set, and 93 *why*-questions against INEX Wikipedia. We discovered that, compared to disjoint windows, using sliding windows results in improved performance of TREC-QA in terms of TDRR, and in improved performance of *why*-QA in terms of success@n and MRR.

1 Introduction

In question answering (QA), text passages are an important intermediary between full documents and exact answers. They form a very natural unit of response for QA systems (Tellex et al., 2003) and it is known from user studies that users prefer answers to be embedded in paragraph-sized chunks (Lin et al., 2003) because they can provide the context of an answer. Therefore, almost all state-of-the-art QA systems implement some technique for extracting paragraph-sized fragments of text from a large corpus.

Most QA systems have a pipeline architecture consisting of at least three components: question analysis, document/passage retrieval, and answer extraction (Hirschman and Gaizauskas, 2001; Voorhees, 2001). The quality of a QA system heavily depends on the effectiveness of the integrated retrieval system (second step of the pipeline): if a retrieval system fails to find any relevant documents for a question, further processing steps to extract an answer will inevitably fail too (Monz, 2003). This motivates the need to study passage retrieval for QA.

There are two common approaches to retrieving passages from a corpus: one is to index each passage as separate document and retrieve them as such. The other option is to first retrieve relevant documents for a given question and then retrieve passages from the retrieved documents. The passages themselves can vary in size and degree of overlap. Their size can be fixed as a number of words or characters, or varying with the semantic content (Hearst and Plaunt, 1993) or the structure of the text (Callan, 1994). The overlap between two adjacent passages can be either zero, in which case we speak of *disjoint passages*, or the passages may be overlapping, which we refer to as *sliding passages*.

In this paper, we compare the effectiveness of several passage retrieval techniques with respect to their usefulness for QA. Our main interest is the contribution of sliding passages as apposed to disjoint passages, and we will experiment with a number of retrieval models. We evaluate the retrieval approaches on two different QA tasks: (1) factoid-QA, as defined by the test collection provided by TREC (Voorhees, 2002; Voorhees, 2003), and (2) a relatively new problem in the QA field: that of answering *why*-questions (*why*-QA).

The remainder of the paper is organized as follows. In the next section, we describe related work on passage retrieval for QA and we motivate what the main contribution of the current paper is. In

Coling 2008: Proceedings of the 2nd workshop on Information Retrieval for Question Answering (IR4QA), pages 26–33
Manchester, UK. August 2008

section 3 we describe our general set-up for passage retrieval in both QA tasks that we consider. In section 4, we present the results of the experiments on TREC-QA data, and in section 5 we present our results on *why*-QA. Section 6 gives an overall conclusion.

2 Related work

The use of passage retrieval for QA has been studied before. For example, (Tellex et al., 2003) performed a quantitative evaluation of passage retrieval algorithms for QA. They compared different passage retrieval algorithms in the context of their QA system. Their system first returns a ranked list of 200 documents and then applies different passage retrieval algorithms to the retrieved documents. They find that the performance of passage retrieval depends on the performance of the pre-applied document retrieval step, and therefore they suggest that document and passage retrieval technology should be developed independently.

A similar message is conveyed by (Roberts and Gaizauskas, 2004). They investigate different approaches to passage retrieval for QA. They identify each paragraph as a seperate passage. They find that the optimal approach is to allow multiple passages per document to be returned and to score passages independently of their source document.

(Tiedemann, 2007) studies the impact of document segmentation approaches on the retrieval performance of IR for Dutch QA. He finds that segmentation based on document structure such as the use of paragraph markup (discourse-based segmentation) works well with standard information retrieval techniques. He tests various other techniques for document segmentation and various passage sizes. In his experimental setting, larger text units (such as documents) produce better performance in passage retrieval. Tiedemann compares different sizes of discourse-based segmentation: sentences, paragraphs and documents. He finds that larger text units result in a large search space for subsequent QA modules and hence reduce the overall performance of the QA system. That is why we do not conduct experiments with different passage sizes in this paper: it is difficult to measure the outcome of such experiments independently of the specific answer extraction system. We adopt Tiedemann's best strategy of document segmentation strategy, i.e., paragraph-based, but with equally sized passages instead.

3 General experiment set-up

The main purpose of our experiments is to study the contribution of sliding windows as apposed to disjoint windows in the context of QA. Therefore, in our experiment setup, we have kept fixed the other segmentation variables, passage size and degree of overlap. We set out to examine two different strategies of document segmentation (disjoint and sliding passages) with a number of retrieval models for two different QA tasks: TREC factoid-QA and *why*-QA.

3.1 Retrieval models

We use the Lemur retrieval engine[1] for passage retrieval because it provides a flexible support for different types of retrieval models including vector space models and language models. In this paper we have selected two vector space models: TFIDF and Okapi BM25 (Robertson and Walker, 1999), and one language model based on Kullback-Leibler (KL) divergence (Lafferty and Zhai, 2001).

The TFIDF weighting scheme is often used in information retrieval. There are several variations of the TFIDF weighting scheme that can effect the performance significantly. The Lemur toolkit provides a variant of the TFIDF model based on the Okapi TF formula (Robertson et al., 1995).

Lemur also provides the implementation of the original Okapi BM25 model, and we have used this model with default values of 1.2 for $k1$, 0.75 for b and 7 for $k3$ as suggested by (Robertson and Walker, 1999). The KL-divergence retrieval model, which implements the cross entropy of the query model with respect to the document model, is a standard metric for comparing distributions, which has proven to work well in IR experiments in the past. To address the data sparseness problem during model estimation, we use the Dirichlet smoothing method (Zhai and Lafferty, 2004) with default parameter values provided in the Lemur toolkit.

Currently, however, the Lemur [2] does not support direct passage retrieval. For these experiments, therefore, we first need to segment documents into passages before indexing them into the

[1] Lemur toolkit: http://www.lemurproject.org

[2] Lemur and Indri are different search engines. Indri provides the $\#passage$ operator, but it doesn't consider paragraph boundaries or sentence boundaries for constructing passages.

Lemur retrieval engine. Our segmenting strategy is explained below.

3.2 Passage identification

For our experiments, we take into account two different corpora: AQUAINT and the Wikipedia XML corpus as used in INEX (Denoyer and Gallinari, 2006). The AQUAINT corpus consists of news articles from the Associated Press, New York Times, and Xinhua News Agency (English version) from 1996 to 2000. The Wikipedia XML collection consists of 659,388 articles as they occured in the online Wikipedia in the summer of 2006. As we have discussed in Section 2, (Tiedemann, 2007) discovered that discourse-based segmentation into paragraphs works well with standard information retrieval techniques. They also observe that larger retrieval units produce better results for passage retrieval, since larger units have higher chance to cover the required information. Therefore, we decide to segment each document into similar sized passages while taking into account complete paragraphs only.

For document segmentation, our method first detects sentences in the text using punctuation marks as separators, and then paragraphs using empty lines as separators. Sentence boundaries are necessary because we aim at retrieving passages that do not contain any broken sentences. The required passages are identified by aligning over paragraph boundaries (merging paragraphs into units until they have the required length ,i.e. 500 characters). The disjoint passages do not share any content with each other, and the sliding passages slide with the difference of one paragraph boundary, i.e., we start forming a new passage from beginning of each paragraph of the document. If paragraph boundaries are not detected, then these sliding passages are half-overlapped with each other.

For the Wikipedia XML corpus, we have found that documents have already been annotated with <p> elements. Thus we consider these elemens as paragraph boundaries instead of empty lines as we did for the AQAINT corpus. We observe that some textual parts of the documents are not covered by the XML paragraph boundaries. Therefore we have extended the existing paragraph boundaries such that the missing text fragments become part of the paragraphs.

We split both corpora into disjoint and sliding windows as we have discussed above. After splitting the 1.03M documents of the AQUAINT-1 collection we have 14.2M sliding passages, and 4.82M disjoint passages. And similarly we got 4.1M sliding passages and 2M disjoint passages from the Wikipedia XML collection of 659,388 documents.

3.3 Evaluation metrics

For our experiments, we use the following metrics for evaluation:

Mean reciprocal rank (MRR) at n is the mean (calculated over all questions) of the reciprocal rank (which is 1 divided by the rank ordinal) of the highest ranked relevant (i.e. answer bearing) passage. RR is zero for a question if no relevant passage is returned by the system at limit n.

Success at n for a question is 1 if the answer to this question is found in top n passages fetched up by our system. Success@n is averaged over all questions.

Total document reciprocal rank (TDRR) (Bilotti et al., 2004) is the sum of all reciprocal ranks of all answer bearing passages per question (averaged over all questions). The value of TDRR is maximum if all retrieved passages are relevant. TDRR is an extension of MRR that favors a system that ranks more that one relevant passage higher than all non-relevant passages. This way, TDRR extends MRR with a notion of recall.

When we compare retrieval performance of two retrieval settings (such as the use of *disjoint* versus *sliding* windows), then we obtain a list of paired scores. That's why we use the Wilcoxon signed-rank test to show the statistical significance of the improvements.

In summary, we experiment with three retrieval models in Lemur: TFIDF, Okapi, and a language model based on the Kullback-Leibler divergence. For each of these retrieval models, we evaluate the use of both sliding and disjoint passages. This makes a total of six retrieval settings.

4 Evaluating passage retrieval for TREC-QA

As test collection for factoid QA, we use a standard set of 822 question/answer pairs from the TREC

QA tasks of 2002-2003. For evaluation of the passage retrieval approaches that we consider, we compute strict scores as defined by (Tellex et al., 2003). Strict scoring means that a retrieved passage is considered relevant if the passage not only matches one of the answer patterns provided by NIST, but its associated document is also listed as one of the relevant documents assessed by NIST. (Bilotti et al., 2004) have reviewed 109 factoid questions of the TREC-2002 task and they have extended the existing set of relevant documents by adding more relevant documents. We have also included this extended list of relevant documents for these questions in our experiment setup.

We evaluate the impact of disjoint and sliding windows on passage retrieval for QA using three different retrieval models, using the MRR@n, Success@n and TDRR@n metrics as described in section 3.3. Table 1 shows the evaluation results (best scores for each measure in bold face). The experiment results show that language model based on Kullback-Leibler divergence shows better performance than two vector space models for both types of windows retrieval according to MRR, success@n and TDRR evaluation metrics.

4.1 Discussion

In a pipeline QA system, the answer extraction module depends on the performance of passage retrieval. If more answer bearing passages are provided in the stream, then there is a high chance of selecting the correct answer from the stream in later stages of QA. (Roberts and Gaizauskas, 2004) have also discussed the importance of this aspect of passage retrieval for QA. They have measured the *answer redundancy* of a retrieval system which measures how many answer bearing passages are returned per question at limit n. (Tiedemann, 2007) have also used this metric and argue that high *redundancy* is desired to make it easier for the answer extraction module to spot possible answers. We consider TDRR as the most important measure for the passage retrieval task since it does not only measure the *redundancy* of a retrieval system but also measures how much improvement there is in returning the relevant passages at top ranks.

According to TDRR@n in table 1, retrieval of sliding windows outperforms retrieval of disjoint windows at all limits of n for all retrieval models. For $n = 100$, the improvement is significant at $p = 0.01$ level. This high value of TDRR@n suggests that segmenting the documents into sliding windows is a better choice in order to return as many relevant passages as possible at top ranks.

If we consider Success@n as evaluation measure instead of TDRR, retrieval of disjoint windows outperforms retrieval of sliding windows. We think that one of the reasons for this behaviour is that since sliding windows overlap with their neighbours, they are more pair-wise similar than disjoint windows. Therefore, it is possible that for some non-answered questions many irrelevant passages are returned at top ranks and that relevant passages are surpressed down.

5 Evaluating passage retrieval for *why*-QA

In the previous section, we showed that for TREC data, the choice of the retrieval model and the type of windows to be retrieved influence on the retrieval performance. We found that for the TREC data, a language modeling approach (based on Kullback-Leibler divergence) on sliding windows gives the best results in terms of TDRR. In this section, we aim to find out what the optimal passage retrieval approach is for a very different type of QA, namely *why*-QA.

5.1 Background of *why*-QA system development

In (Verberne et al., 2008), we present an approach for *why*-QA that is based on paragraph retrieval from the INEX Wikipedia corpus (Denoyer and Gallinari, 2006). Our system for *why*-QA consists of two modules: a passage retrieval module and a re-ranking module. In earlier retrieval experiments, we used the Wumpus retrieval system (Buttcher, 2007), and we defined passages simply by the XML paragraph markup <p>. Passage ranking in Wumpus is done by the QAP passage scoring algorithm (Buttcher et al., 2004).

The second module of our *why*-system is a re-ranking step that uses syntactic features of the question and the retrieved answers for adapting the scores of the answers and changing the ranking order. The weights of the re-ranking features have been optimized by training on our question answer data in five folds[3] using a genetic algorithm. We let Wumpus retrieve and rank 150 paragraphs per

[3] In five turns, we tune the feature weights on four of the five folds and evaluate them on the fifth

Table 1: Results for passage retrieval for TREC-QA using disjoint windows (DW) and sliding windows (SW). ** indicates a significant improvements of sliding windows over disjoint windows at the $p = 0.01$ level.

		MRR		Success@n		TDRR	
n	retrieval model	DW	SW	DW	SW	DW	SW
10	TFIDF	0.327	0.326	51.8%	50.1%	0.465	0.637
	Okapi	0.322	0.328	51.9%	51.2%	0.459	0.649
	KL	**0.355**	0.345	**55.7**%	51.3%	0.518	**0.710**
100	TFIDF	0.336	0.386	54.1%	53.3%	0.517	0.819**
	Okapi	0.333	0.339	**77.0**%	76.2%	0.535	0.835**
	KL	**0.363**	0.353	**77.1**%	75.2%	0.525	**0.902****

question. This number of 150 answers was chosen as a trade-off between covering as many as possible of the relevant answers retrieved by Wumpus, and the system load that was needed for automatic syntactic analysis of all answers in the second (re-ranking) module of the system. For evaluation of the results, we performed manual assessment of all answers retrieved, starting at the highest-ranked answer and ending as soon as we encountered a relevant answer[4].

The results for our original *why*-system are in Table 2. We show the results in terms of success@n and MRR@n. As opposed to the evaluation of TREC-QA, we do not consider TDRR as evaluation measure for experiments on *why*-QA. This is because in *why*-QA, we are only interested in the top-ranked answer-bearing passage. For calculating TDRR, assessment of all 150 retrieved answers would be necessary.

Table 2 shows that success@150 for the retrieval module (Wumpus/QAP) is 73.1%. This means that for 26.9% of the questions, no relevant answer is retrieved in the first module. Re-ranking the answers cannot increase MRR for these questions, since none of the 150 answers in the result list is relevant. We consider a success@150 score of 73.1% to be quite low. We aim to improve the performance of our system by optimizing its first module, passage retrieval.

We experiment with a number of passage retrieval approaches in order to reach better retrieval in the first module of our system. We aim to find out which type of retrieval model and what window type (disjoint or sliding) gives optimal results for retrieving passages relevant to *why*-questions. If the retrieval performance indeed goes up, we will apply our re-ranking module to the newly retrieved data to see what overall system performance we can reach with the new retrieval approach.

[4]We didn't need to assess the tail since we were only interested in the highest-ranked relevant answer for calculating MRR and success@n

5.2 Data and evaluation setup

For development and testing purposes, we use the Webclopedia question set by (Hovy et al., 2002). This set contains questions that were asked to the online QA system `answers.com`. 805 of these questions are *why*-questions. We manually inspect a sample of 400 of the Webclopedia *why*-questions. Of these, 93 have an answer in the Wikipedia XML corpus (see section 3). Manual extraction of one correct answer for each of these questions results in a set of 93 *why*-questions and their reference answer.

In order to be able to do fast evaluation of the different evaluation settings, we manually create an answer pattern for each of the questions in our set. These answer patterns are based on a set of 93 reference answers (one answer per question) that we have manually extracted from the Wikipedia corpus. An answer pattern is a regular expression that defines which of the retrieved passages are considered a relevant answer to the input question.

As opposed to the answer patterns provided by NIST for the evaluation of factoid QA (see section 4), our answer patterns for *why*-questions are relatively strict. A *why*-answer can be formulated in many different ways with different words, which may not all be in the answer pattern. For a factoid question such as "When was John Lennon born?", the answer is only one phrase, and the answer pattern is short and unambiguous, i.e. `/1940/`. However, if we consider the *why*-question "Why are some organ transplants unsuccessful?", the answer pattern cannot be stated in one phrase. For

Table 2: Results for the original *why*-QA pipeline system

	success@10	success@150	MRR@150
Wumpus/QAP Retrieval	43.0%	73.1%	0.260
+ Re-ranking module	54.8%	73.1%	0.380

this example, we created the following answer pattern based on the pre-extracted reference answer[5]: `/.*immune system.*foreign tissues.*destroy.*/`. It is however possible that a relevant answer is formulated in a way that does not match this regular expression. Thus, the use of answer patterns for the evaluation of *why*-QA leads to conservative results: some relevant answers may be missed in the evaluation procedure.

After applying the answer patterns, we count the questions that have at least one relevant answer in the top 10 and the top 150 of the results (success@10, success@150). For the highest ranked relevant answer per question, we determine the reciprocal rank (RR). If there is no correct answer retrieved by the system at $n = 150$, the RR is 0. Over all questions, we calculate the MRR@150.

5.3 Passage retrieval results

We segment and index the Wikipedia corpus as described in section 3 and run all six retrieval settings on our set of 93 *why*-questions. For consistent evaluation, we applied the answer patterns that we created to the newly retrieved Lemur data as well as to the original Wumpus output.

The retrieval results for all settings are in Table 3. We show both success@10 and success@150, and MRR@150 for each setting. Success@150 is important if we consider the current results as input for the re-ranking module. As explained before, re-ranking can only be successful if at least one relevant answer is retrieved by the retrieval module. For each measure (s@10, s@150 and MRR@150), the score of the highest-scoring setting is printed in bold face.

As expected, the evaluation of the Wumpus data with the use of answer patterns gives somewhat lower scores than evaluation based on manual assessment of all answers (table 2). This confirms our idea that the use of answer patterns for *why*-QA leads to conservative results. Thus we can state that the Lemur scores shown in table 3 are not overestimated and therefore reliable.

Since we are using the output of the passage retrieval module as input for our re-ranking module, we are mainly interested in the scores for success@150. For the four retrieval models, we see that TFIDF seems to score somewhat better on retrieving sliding windows in terms of success@150 than Okapi and the Kullback-Leibler language model. On the other hand, Kullback-Leibler and QAP seem to perform better on retrieving disjoint windows. However, these differences are not significant at the $p = 0.01$ level. For the differences between disjoint and sliding windows for all retrieval models together, we see that retrieval of sliding windows gives significantly better results than disjoint windows in terms of success@150 ($p < 0.001$).

5.4 The influence of passage retrieval on our pipeline system

As described in section 5.1, our system is a pipeline: after passage retrieval, we apply a re-ranking module that uses syntactic information for re-scoring the results from the retrieval module. As input for our re-ranking module we use the output of the retrieval setting with the highest success@150 score: Lemur/TFIDF on sliding windows. For 81.7% of the questions in our set, Lemur/TFIDF retrieved an answer in the top-150. This means that the maximum success@10 score that we can obtain by re-ranking is 81.7%.

For weighting the feature values, we re-use the weights that we had earlier found from training on our set of 93 questions and the 150 answers that were retrieved by Wumpus. We again take into account five-fold cross validation for evaluation. For a detailed description of our re-ranking module and the syntactic features that we exploit, we refer to (Verberne et al., 2008).

The results from re-ranking are in Table 4. In the table, four system versions are compared: (1) the original Wumpus/QAP module, (2) the original *why*-pipeline system: Wumpus/QAP with re-ranking, (3) TFIDF-sliding and (4) the new

[5]The pre-extracted reference answer is: "This is because a normal healthy human immune system can distinguish foreign tissues and attempts to destroy them, just as it attempts to destroy infective organisms such as bacteria and viruses."

Table 3: Results for passage retrieval on *why*-questions against Wikipedia using disjoint windows (DW) and sliding windows (SW)

	Success@10		Success@150		MRR@150	
Retrieval model	DW	SW	DW	SW	DW	SW
Baseline: Wumpus/QAP	40.9%		72.0%		0.229	
Lemur/TFIDF	43.0%	45.2%	71.1%	**81.7**%	0.247	**0.338**
Lemur/Okapi	41.9%	44.1%	67.7%	79.6%	0.243	0.320
Lemur/KL	48.9%	**50.0**%	72.8%	77.2%	0.263	0.324

pipeline system: TFIDF-sliding with re-ranking. We again show MRR, success@10 and success@150. For each measure, the score of the highest-scoring setting is printed in bold face.

After applying our re-ranking module (right bottom setting), we find a significant improvement over bare TFIDF (left bottom setting). In terms of MRR, we also see an improvement over the results that we had obtained by re-ranking the Wumpus/QAP output (right top setting). However, success@10 does not show significant improvement. The improvement that the re-ranking module gives is smaller for the TFIDF retrieval results (MRR goes from 0.338 to 0.359) than for the QAP results (MRR increases from 0.260 to 0.328). We suspect that this may be due to the fact that we used feature weights for re-ranking that we had earlier obtained from training on the Wumpus/QAP data (see section 5.4). It would be better to re-train our feature weights on the Lemur data. Probably, re-ranking can then make a bigger contribution than it does now for the Lemur data.

6 Overall conclusion

In this paper we have investigated the contribution of sliding windows as apposed to disjoint windows with different retrieval modules for two different QA tasks: the TREC-QA 2002–2003 task and *why*-QA.

For the TREC factoid-QA task, we have found that retrieval of sliding windows outperfoms retrieval of disjoint windows in returning as many relevant passages as possible on top ranks (according to the TDRR metric). The experimental results show that a language model based on Kullback-Leibler divergence gives better performance than two vector space models for both types of windows retrieval according to MRR, success@n and TDRR evaluation metrics. We found that the number of answered questions (success@n) was slightly lower when we used sliding windows for passage retrieval than disjoint windows, but we think one of the reasons is that sliding windows are more homogeneous than disjoint windows, and therefore for some questions more irrelevant passages are returned at top ranks and relevant passages are surpressed down.

For the task of retrieving answers to *why*-questions from Wikipedia data, we found that the best retrieval model is TFIDF, and sliding windows give significantly better results than disjoint windows. We also found better performance for our complete *why*-pipeline system after applying our existing re-ranking module to the passages retrieved with TFIDF-sliding.

In general, we find that for QA, sliding windows give better results than disjoint windows in the passage retrieval step. The best scoring retrieval model depends on the task under consideration, because the nature of the documents and question sets differ. This shows that for each specific QA task, different retrieval models should be considered.

In the future, we aim to boost passage retrieval for QA even more by applying query expansion techniques that are specific to the QA tasks that we consider, i.e. TREC factoid-QA and *why*-QA.

References

Bilotti, M.W., B. Katz, and J. Lin. 2004. What works better for question answering: Stemming or morphological query expansion. In *Proceedings of the SIGIR 2004 Workshop IR4QA: Information Retrieval for Question Answering, July*.

Buttcher, S., C.L.A. Clarke, and G.V. Cormack. 2004. Domain-specific synonym expansion and validation for biomedical information retrieval (multitext experiments for trec 2004).

Buttcher, S. 2007. The wumpus search engine. http://www.wumpus-search.org/.

Callan, James P. 1994. Passage-level evidence in document retrieval. In *SIGIR*, pages 302–310.

Table 4: Results for the *why*-QA pipeline system for best-scoring passage retrieval setting compared against the Wumpus baseline, for both bare retrieval and the complete system with re-ranking

	Success@10		Success@150		MRR	
Retrieval model	Bare	+Re-rank	Bare	+Re-rank	Bare	+Re-rank
Baseline: Wumpus/QAP-disjoint	43.0%	54.8%	73.1%	73.1%	0.260	0.328
Lemur/TFIDF-sliding	45.2%	**55.9**%	81.7%	81.7%	0.338	**0.359**

Denoyer, L. and P. Gallinari. 2006. The Wikipedia XML corpus. *ACM SIGIR Forum*, 40(1):64–69.

Hearst, Marti A. and Christian Plaunt. 1993. Subtopic structuring for full-length document access. In *ACM-SIGIR, 1993*, pages 59–68.

Hirschman, L. and R. Gaizauskas. 2001. Natural language question answering: the view from here. *Nat. Lang. Eng.*, pages 275–300.

Hovy, E.H., U. Hermjakob, and D. Ravichandran. 2002. A question/answer typology with surface text patterns. In *Proceedings of the Human Language Technology conference (HLT)*, San Diego, CA.

Lafferty, J. and C. Zhai. 2001. Document language models, query models, and risk minimization for information retrieval. In *In Proceedings of SIGIR'01*, pages 111–119.

Lin, J., D. Quan, V. Sinha, K. Bakshi, D. Huynh, B. Katz, and D.R. Karger. 2003. The role of context in question answering systems. *Conference on Human Factors in Computing Systems*, pages 1006–1007.

Monz, Christof. 2003. Document retrieval in the context of question answering. In *ECIR*, pages 571–579.

Roberts, I. and R. Gaizauskas. 2004. Evaluating passage retrieval approaches for question answering. In *In Proceedings of ECIR , 2004.*

Robertson, Stephen E. and Steve Walker. 1999. Okapi/keenbow at trec-8. In *Text Retrieval Conference*.

Robertson, Stephen E., Steve Walker, Micheline Hancock-Beaulieu, and Gatford M. 1995. Okapi at trec-3. In *Text Retrieval Conference*, pages 109–26.

Tellex, S., B. Katz, J. Lin, A. Fernandes, and G. Marton. 2003. Quantitative evaluation of passage retrieval algorithms for question answering. In *In SIGIR conference on Research and development in informaion retrieval, 2003*, pages 41–47.

Tiedemann, Jörg. 2007. Comparing document segmentation strategies for passage retrieval in question answering. *In Proceedings of RANLP 07, Borovets, Bulgaria*.

Verberne, Suzan, Lou Boves, Nelleke Oostdijk, and Peter-Arno Coppen. 2008. Using Syntactic Information for Improving Why-Question Answering. In *Proceedings of The 22nd International Conference on Computational Linguistics (COLING 2008)*.

Voorhees, Ellen. 2001. Overview of trec 2001 question answering track. In *In Proceedings of TREC*.

Voorhees, Ellen. 2002. Overview of trec 2002 question answering track. In *In Proceedings of TREC*.

Voorhees, Ellen. 2003. Overview of trec 2003 question answering track. In *In Proceedings of TREC*.

Zhai, ChengXiang and John D. Lafferty. 2004. A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst.*, pages 179–214.